

## SYSTEMS AND METHODS FOR PERFORMING MEMORY COMPRESSION

### CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation of U.S. patent application Ser. No. 15/663,115, entitled “SYSTEMS AND METHODS FOR PERFORMING MEMORY COMPRESSION”, filed Jul. 28, 2017, the entirety of which is incorporated herein by reference.

### BACKGROUND

#### Technical Field

[0002] Embodiments described herein relate to the field of computing systems and, more particularly, to efficiently moving data for storage and processing.

#### Description of the Related Art

[0003] Generally speaking, a variety of computing systems include a processor and a memory, and the processor generates access requests for instructions and application data while processing one or more software applications. When fetching instructions and data, the processor checks a hierarchy of local cache memories and, if not found, the processor issues requests for the desired instructions and data to main memory or other storage such as, a CD-ROM, or a hard drive, for example.

[0004] At times, the number of software applications simultaneously running on the computing system reaches an appreciable number. In addition, a variety of computing systems include multiple processors such as a central processing unit (CPU), data parallel processors like graphics processing units (GPUs), digital signal processors (DSPs), and so forth. Therefore, the amount of instructions and data being used for processing the multiple software applications appreciably grows. However, the memory storage locations in the local cache memories have a limited amount of storage space. Therefore, swapping of the instructions and data between the local cache memories and the persistent storage occurs.

[0005] The swapping and corresponding latency for waiting for requested information to be loaded reduces performance for the computing system. To reduce an amount of storage for a particular quantity of data, the data is compressed. Such compression takes advantage of repeated sequences of individual data bits included in the data. When the data is to be accessed, the data is decompressed, and then possibly re-compressed once the access has been completed.

[0006] Generally speaking, when a general-purpose processor, such as a central processing unit (CPU), is performing a software routine to compress and/or decompress data, it is occupied for the duration of the operations. Additionally, in a system that includes multiple processors, many times, the CPU is the only processor with support for retrieving, compressing and decompressing the desired data. Therefore, the CPU is partially or fully unavailable while performing one or more of local and network data retrieval and compression. Further, the other processors incur delays while waiting for the CPU to finish the retrieving, compressing and decompressing operations on their behalf.

[0007] In view of the above, methods and mechanisms for efficiently moving data for storage and processing are desired.

### SUMMARY

[0008] Systems and methods for efficiently moving data for storage and processing are contemplated. In various embodiments, a computing system includes a memory, a cache memory and a processor. In response to receiving a compression instruction, the processor fetches data from the memory into the cache memory. In some embodiments, the data is partitioned into multiple input words. Following, the processor loads multiple input words from the cache memory into a read buffer within the processor. A compression unit within the processor includes circuitry for executing the compression instruction. Therefore, the processor is available for processing other operations while the compression unit processes the compression instruction.

[0009] In an embodiment, the compression unit selects two or more input words of the multiple words to be used as assigned input words. The compression unit includes multiple hardware lanes for performing operations of a compression algorithm. Each of the two or more hardware lanes of the multiple hardware lanes are assigned to a respective one of the selected two or more input words. Each of the two or more hardware lanes generates a respective compressed packet based on at least its assigned input word. To generate a compressed packet, each hardware lane uses a value to compare against the assigned word to determine intra-group dependencies. However, in various embodiments, prior to determining intra-group dependencies of a first group of words with a same index, a dictionary is accessed for a younger second group of words, each word in the second group having a same index.

[0010] The compression unit combines the compressed packets into a group of compressed packets. In some embodiments, the compression unit further combines two or more groups into a packed group and writes the packed group into a write buffer. At a later time, the processor sends the packed group from the write buffer to a target storage location.

[0011] In various embodiments, as each assigned input word is processed, it is searched for repeated sequences of data bits by being compared against previously seen data. In some embodiments, the previously seen data is stored in entries of a data structure (e.g., such as a table) referred to as a dictionary. In some embodiments, the multiple hardware lanes perform steps of a combination of a statistical-based compression algorithm and a dictionary-based compression algorithm. In some embodiments, each of the two or more selected input words, which are assigned to the two or more hardware lanes, has a corresponding index pointing to a same entry of the multiple entries of the table. In some embodiments, the contents of the same entry of the table are read from the dictionary once for processing of the input words currently assigned to the two or more hardware lanes. In an embodiment, the hardware lane with the oldest input word of the two or more assigned input words generates the single read request. Additionally, the hardware lane with the youngest input word of the two or more assigned input words generates the single write request for updating the table upon completion of the compression of the two or more assigned input words. Therefore, the multiple read and write requests for the sequences stored in the table for a serial